ISSN 1410 - 9158

Majalah Ilmiah
Informatika, Komputer, dan Bisnis

FORMAT

Volume 6, Nomor 3, September 2005

ANALISA DAN DESAIN BERORIENTASI OBYEK PENGAJUAN JABATAN FUNGSIONAL DOSEN PERGURUAN TINGGI,
Endang Wahyuningsih

KLASIFIKASI TERHADAP DATA SUATU OBYEK DENGAN ANALISIS DISKRIMINAN,
Erna Hudianti Pujiarini

ANALISIS PENGARUH PENGUMUMAN LABA TAHUNAN TERHADAP RETURN SAHAM,
Aloysius Agus Subagyo

SMS POLLING,
Indra Yatini B.

PERANCANGAN SISTEM INFORMASI AKADEMIK BERORIENTASI OBYEK,
Cuk Subiyantoro

MEMBANGUN SISTEM LINUX TANPA PARTISI LINUX,
Wagito

PERANGKAT LUNAK BANTU UNTUK IMPLEMENTASI DIAGRAM FSA,
Febri Nova Lenti

ORIENTASI KEWIRAUSAHAAN,
Dison Librado

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER AKAKOM
YOGYAKARTA

Majalah Ilmiah
Informatika, Komputer, dan Bisnis
FORMAT

PELINDUNG:
Drs. Agus Sulistya Pribadi, S.H., M.M.

KETUA UMUM:
Drs. G.P. Daliyo, Dipl. Com.

KETUA DEWAN REDAKSI:
Bambang P.D.P., S.E., Akt., S.Kom., M.MSi.

ANGGOTA DEWAN REDAKSI:
Ir. F. Soesianto, B.Sc.E., Ph.D.
Prof. H. Adhi Susanto, M.Sc., Ph.D.
Drs. Tri Prabawa, M.Kom.
Ir. Surjono, M.Phil.
Ir. Sudarmanto, M.T.
Ir. M. Guntara, M.T.
Ir. Totok Suprawoto, M.M.
Budi Sugihardjo, S.E., M.M.
Heru Agus Triyanto, S.E., M.M.

REDAKTUR PELAKSANA:
Indra Yatini Buryadi, S.Kom., M.Kom.

SEKRETARIS:
Al. Agus Subagyo, S.E.

LAYOUT dan PRODUKSI:
Dison Librado, S.E.

SIRKULASI:
Totok Budioko, S.T.

DOKUMENTASI:
Dra. Torsinawati
Sukar

Majalah Ilmiah FORMAT diterbitkan empat bulan sekali oleh
STMIK AKAKOM dengan ISSN 1410 - 9158

Pendapat yang dinyatakan dalam majalah ini
adalah sepenuhnya pendapat pribadi
Segala sesuatu yang berhubungan dengan penerbitan majalah dapat disampaikan secara
tertulis maupun lisan kepada redaksi

ALAMAT REDAKSI:
STMIK AKAKOM
Jl. Raya Janti, *Ring Road* Timur, Yogyakarta 55198
Telepon : 62-0274-486664
Faksimile : 62-0274-486438 E-mail : format@netexecutive.com

Dari Redaksi

Kami panjatkan puji dan syukur atas rahmat dan berkah dari Tuhan Yang Maha Esa hingga kami dapat menyelesaikan dan menerbitkan majalah Format pada nomor pertama, tahun pertama. Pada tahun yang pertama ini kami mencoba untuk memperluas cakupan materi dari berbagai hasil penelitian dan karya ilmiah, namun tetap sesuai dengan misinya.

Dalam edisi ini para pembaca akan melihat topik-topik mengenai *Analisa dan Desain Berorientasi Obyek Pengajuan Jabatan Fungsional Dosen Perguruan Tinggi, Klasifikasi Terhadap Data suatu Obyek dengan Analisis Diskriminan, Analisis Pengaruh Pengumuman Laba Tahunan terhadap Return Saham, SMS Polling, Perancangan Sistem Informasi Akademik Berorientasi Obyek, Membangun Sistem Linux tanpa Partisi Linux, Perangkat Lunak Bantu untuk Implementasi Diagram FSA, Orientasi Kewirausahaan,* yang sekiranya akan menarik untuk diulas.

Harapan kami semoga apa yang kami suguhkan kali ini dapat membawa manfaat bagi peminat, dan menambah referensi pembaca pada bidang-bidang tertentu. Terima kasih diucapkan, atas saran dan masukan yang telah kami terima demi kemajuan majalah ilmiah ini. Saran, ide, dan gagasan dari para pembaca tetap kami tunggu untuk perbaikan pada penerbitan edisi mendatang di abad millenium ini.

Daftar Isi

Analisa dan Desain Berorientasi Obyek Pengajuan Jabatan Fungsional Dosen Perguruan Tinggi
Endang Wahyuningsih 639
Klasifikasi Terhadap Data suatu Obyek dengan Analisis Diskriminan
Erna Hudianti Pujiarini 659
Analisis Pengaruh Pengumuman Laba Tahunan terhadap Return Saham
Aloysius Agus Subagyo 681
SMS Polling
Indra Yatini B 697
Perancangan Sistem Informasi Akademik Berorientasi Obyek
Cuk Subiyantoro 731
Membangun Sistem Linux tanpa Partisi Linux
Wagito 741
Perangkat Lunak Bantu untuk Implementasi Diagram FSA
Febri Nova Lenti 759
Orientasi Kewirausahaan
Dison Librado 777

PERANGKAT LUNAK BANTU UNTUK IMPLEMENTASI DIAGRAM FSA

Oleh
Febri Nova Lenti

ABSTRAK

Pada beberapa masalah spesifik yang berkaitan dengan keputusan dan model mesin hanya tepat jika solusinya didasarkan pada solusi automata. Beberapa contoh masalah tersebut seperti mesin jaja, lift, mesin atm, protokol komunikasi data, mesin modulo, kompilator dan lain sebagainya akan sangat tepat jika dimodelkan dengan menggunakan automata, karena akan menolong kita untuk memahami perilaku / watak dari sistem yang akan dibangun. Mengingat bahwa teori automata dapat diterapkan untuk memodelkan suatu sistem maka perlu dibangun suatu perangkat lunak bantu (*tool*) yang akan mengkonstruksi diagram automata dari suatu defenisi formalnya.Pada pengembangan perangkat lunak bantu ini menggunakan struktur data sederhana yaitu record, array statis dan enumerasi. Alasan pemilihan struktur ini adalah karena akan membuat pemakaian memori menjadi efektif dan efisien sesuai dengan kebutuhan sistem.

Keywords : FSA, DFA, NFA, diagram, diagram transisi, record, array, enumerasi

PENDAHULUAN

Penerapan teoritis automata untuk pengembangan suatu sistem adalah dengan menggunakan teori automata sebagai sebuah paradigma yang menggabungkan spesifikasi sistem, verifikasi dan sintesis. Masalah-masalah yang berkaitan dengan keputusan dan sintesis beberapa diantaranya didasarkan pada solusi automata dan tidak ada solusi lain yang diketahui dapat diterapkan padanya.

Suatu teori hanya menarik bila itu membantu dalam mencari solusi terbaik. Teori memberikan konsep dan prinsip yang menolong untuk memahami 'perilaku' dari suatu disiplin ilmu. Untuk mempelajari prinsip-prinsip dasar inilah kita mengkonstruksi model abstrak dari komputer dan komputasi. Meskipun model tersebut terlalu sederhana untuk diterapkan langsung pada dunia nyata, keuntungan yang diperoleh untuk mempelajarinya adalah memberikan landasan bagaimana suatu pengembangan didasarkan.

Oleh karena itu dibutuhkan suatu alat bantu (*tool*) yang akan membantu memodelkan diagram automata.

1 Teori Automata

1.1 Automata Berhingga

Automata adalah suatu mesin sekuensial (otomatis), yang menerima input (dari pita masukan) dan mengeluarkan output, keduanya dalam bentuk diskrit. Automata mempunyai sifat-sifat :

- Kelakuan mesin bergantung pada rangkaian masukan yang diterima mesin tersebut.
- Setiap saat, mesin dapat berada pada satu status tertentu dan dapat berpindah ke status baru karena adanya perubahan input.
- Rangkaian input (diskrit) pada mesin automata dapat dianggap sebagai **bahasa** yang harus "dikenali" oleh sebuah automata. Setelah pembacaan input selesai, mesin automata kemudian membuat "keputusan".

Jenis- jenis automata :

Jenis	Pita masukan	Arah Head	Memori
Finite *State*	Read Only	1 arah	-
Push Down	Read Only	1 arah	stack
Linear-Bounded	R/W	2 arah	(bounded)
Turing Machine	R/W	2 arah	(unbounded)

Pada bahasan ini jenis automata yang akan dipakai adalah *Finite State Automata (FSA). FSA* adalah mesin yang dapat mengenali kelas bahasa reguler dan memiliki sifat-sifat :

1. Pita masukan (*input tape)* berisi rangkaian simbol (string) yang berasal dari himpunan simbol / alfabet.
2. Setiap kali setelah membaca satu karakter, posisi *read head* akan berada pada simbol berikutnya.
3. Setiap saat, *FSA* berada pada status tertentu
4. Banyaknya status yang berlaku bagi *FSA* adalah berhingga.

FSA sendiri mempunyai bermacam-macam jenis yaitu :

- Automata berhingga yang tidak deterministik / *Non-deterministic Finite Automata (NFA)*
- Automata berhingga yang deterministik / *Deterministic Finite Automata (DFA)*
- *Non deterministic Finite Automata* dengan μ transisi
- *Finite State Transducer* (M*oore* dan *Mealy*)

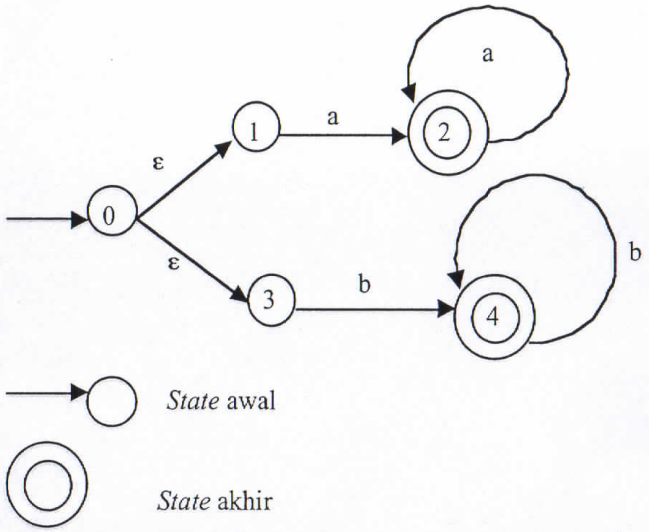
1.1.1 Automata Berhingga Non Deterministik *(NFA)*

Suatu automata berhingga yang tidak deterministik adalah model matematika yang terdiri dari

1. Himpunan dari *state* yang dinyatakan oleh S.
2. Himpunan simbol input d (alfabet dari simbol input).
3. Fungsi transisi yang memetakan pasangan *state*– simbol ke dalam himpunan *state*.
4. *State* S_0 yang disebut dengan *state* awal.
5. Himpunan dari *state* S sebagai *state* penerima akhir.

Suatu *NFA* dapat direpresentasikan dalam bentuk bagan sebagai suatu graf yang diberi label dan disebut dengan graf transisi. Dalam graf transisi ini nodal adalah

state dan label dari sisi menyatakan fungsi transisi, contoh Graf transisi NFA dapat dilihat pada gambar 1.



Gambar 1. NFA *penerima aa'| bb'*

Gambar 1. diatas mempunyai diagram transisi sebagai berikut :

δ	A	B
0	∅	∅
1	2	∅
2	2	∅
3	∅	4
4	∅	4

Disamping itu *NFA* diatas mengandung *e-move*, (e berarti *empty*) yang artinya dapat merubah keadaan/ *state* tanpa membaca input. Pada gambar 1. diatas *state* 0 dapat berpindah ke *state* 1 atau *state* 3 tanpa membaca input.

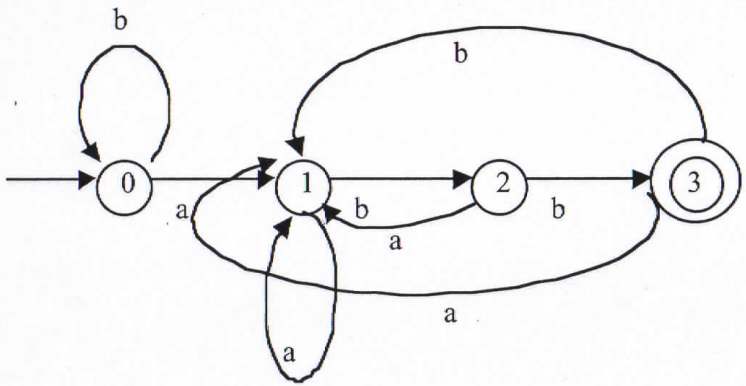
Selanjutnya bahasa-bahasa yang diterima oleh suatu automata berhingga bisa dinyatakan secara sederhana dengan ekspressi regular (*Regular Expression / RE*). *RE* memberikan suatu pola atau template untuk untai/ string dari suatu bahasa. *RE* pada gambar diatas adalah aa'| bb'. * artinya dapat diulang mulai 0 - n kali, dan | berarti "atau".

1.1.2 Automata Berhingga Deterministik (*DFA*)

Suatu automata berhingga yang deterministik merupakan hal khusus dari suatu automata berhingga yang tidak deterministik di mana :

a. Tidak ada *state* yang mempunyai peralihan e, artinya tidak ada peralihan atas input e.
b. Untuk setiap *state* S dan simbol input d, ada paling banyak satu sisi yang meninggalkan s.

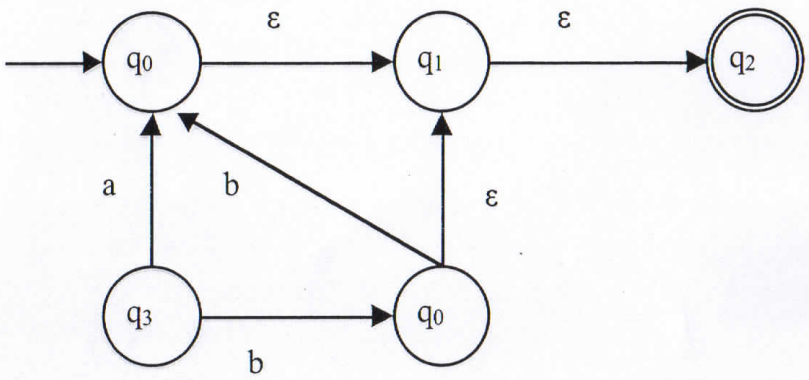
Contoh DFA dapat dilihat pada gambar 2



Gambar 2 DFA *yang menerima (a|b)'abb*

1.1.3 Automata Non Deterministik dengan ε Transisi

Pada *non-deterministic Finite Automata* dengan ε transisi, diperbolehkan merubah *state* tanpa membaca *input*. Disebut dengan transisi ε karena tidak bergantung pada suatu input ketika melakukan transisi. (Gambar 3)



Gambar 3 Mesin NFA *dengan ε transisi*

Dari Gambar 3 dapat dilihat bahwa dari q0 tanpa membaca input dapat berpindah ke q1, dari q1 tanpa membaca input dapat berpindah ke q2, dan dari q4 tanpa membaca input dapat berpindah ke q1. Salah satu kegunaan dari transisi ε ini memudahkan untuk mengkombinasikan FSA.

Pada NFA ε transisi ini dikenal suatu defenisi yang disebut ε-*closure*. E-*closure* ini adalah himpunan status-status yang dapat dicapai dari suatu status tanpa membaca input. Dari gambar 3 dapat dilihat ε-*closure*(q0)={q0, q1, q2), artinya dari status q0 tanpa membaca input dapat mencapai status q0, q1, dan q2.

2. STRUKTUR DATA

Yang dimaksud dengan struktur data adalah bagaimana data disimpan sehingga program dapat memanipulasi data. Beberapa varian struktur data sebagai berikut:

2.1 ARRAY STATIS

Array adalah sekumpulan elemen bertype sama, yang mempunyai sebuah nama (nama array) dan setiap elemen dapat diacu melalui indeksnya. Array dengan satu indeks disebut array berdimensi satu, vektor, larik atau label. Array dengan dua indeks disebut array dua dimensi atau matriks. Array dapat mempunyai dimensi lebih dari dua. Yang harus diperhatikan adalah :

- nama array (seluruh elemen). Dalam bahasa C nama array mengacu ke elemen yang ke-0
- dimensi array (banyaknya indeks)
- ukuran array, atau batas nilai indeks. Dalam bahasa C, batas minimum nilai indeks selalu 0

Array dapat didefinisikan secara statik atau secara dinamik. Array statik adalah array yang ukurannya ditentukan saat kompilasi. Sedangkan array dinamik adalah array yang ukurannya didefinisikan pada saat run time dengan perintah alokasi memori. Perhatikan presedensi operator dalam mengakses elemen dengan array. Urutan akses array adalah *"row column order"*.

Array statik

- Definisi statik:

<type> <nama> [<uk-l>][<uk-2>]...

- Contoh definisi array statik:

```
float TabX[200] ;
int MatA[3] [3] ;
```

Indeks dan ukuran array:

<uk-n> menyatakan ukuran array di dimensi ke n

dengan indeks selalu mulai dari 0 hingga <uk-n>-l.

Isi/nilai dari <nama> sama dengan &<nama>[0],

*(<nama>+l) sama dengan <nama>[l],dst.

- Contoh deklarasi, inisialisasi dan cara mengacu:

Deklarasi:	Cara mengacu:
float TabX[200];	TabX[3]
int MatA[3][3];	MatA[0][0]

- Inisialisasi array statik adalah per baris, contoh:

```
int a[3][4] = ( (1,7,1,2}, (1,9,3,9), (1,6,0,1) );
```

2.2 RECORD

Struktur merepresentasikan suatu type komposisi dalam konsep algoritmik, yaitu sebuah type yang terdiri dari komponen-komponen bertype tertentu (yang tentunya boleh rekursif, yaitu bertype seperti definisi type tersebut)

Deklarasi/definisi:

```
Struct (<nama struktur>] {
/* definisi komponen Struktur */
        <type-I> <anggota-I>;
        <type-2> <anggota-2>;
        <type-n> <anggota-n>;
        } [<nama variabel struktur>] ;
```

Ada dua cara untuk mengakses elemen **struktur :**

- Pengaksesan elemen dengan operator .

```
<nama variabel struktur>.<anggota>
```

- Pengaksesan lewat pointer:

```
<pointer ke struktur> -> <anggota:>
(* <pointer ke struktur>).<anggota>
```


Contoh pendefinisian dan akses struktur komposisi

Berikut ini diberikan tiga buah contoh potongan deklarasi program yang "sama" efeknya, namun "berbeda" pada prinsipnya. Perhatikan baik-baik, nama nama type, nama variabel, nama "tag".

Contoh I: definisi nama variabel dengan struktur komposisi yang terdiri dari nama, kelas, beratbadan dan sekaligus mendefinisikan variabel Rocky, Mike Tyson, Ali sebagai struktur yang mempunyai elemen nama, kelas, beratbadan :

```
/* Berikut ini adalah definisi variabel ber struktur koroposisi */
  struct
    (
     char nama[20] ;
       char kelas[10] ;
    float beratbadan;
     } Rocky, MikeTyson, Ali;
```

Contoh Cara akses: Jika Rocky bertipe struct Petinju, maka:

```
/* ingat, assignment ke string tidak dibolehkan */
    strcpy(Rocky.nama, "Rocky Balboa");
    strcpy(Rocky.kelas, "Berat");
    Rocky.beratbadan = 98.0;
```

Seringkali, untuk struktur yang rekursif, salah .satu teknik adalah dengan memanfaatkan *tag*. Contoh struktur rekursif *(node* pohon biner): pada contoh ini tnode adalah "tag", yaitu "nama" yang diberikan untuk menggantikan sekumpulan/field di antara {
dan }

```
struct tnode {
                char* nama; /* nama node */
                struct tnode* kiri; /* anak kiri */
```

```
        struct tnode* kanan; /* anak kanan */
        } node ;
    struct tnode *left; /* variabel left :pointer ke sebuah tnode
```

2.3 ENUMERASI

Elemen tipe enumerasi merupakan konstanta bertipe integer, dengan harga mulai dari 0, dengan progresi satu untuk harga berikutnya. Pemrogram dapat menentukan harga ini. Jika ada harga yang telah ditetapkan, maka harga berikutnya jika tidak ditetapkan merupakan progresi dari harga terakhir yang ditetapkan. Contoh definisi type dasar :

```
int j; /* hanya deklarasi */

float ff; /* hanya deklarasi */

        float rf=0.0; /* deklarasi dan inisialisasi */

int i==2;      /* deklarasi dan inisialisasi */

enum numeral {nol, two, three, four, five) numero;

`/* nol = 0, two = I, dst */

  enum { Centaurus, Crux, Scorpio, Lupus, Pegasus, Perseus. Andromeda=31,
  Cassiopea } RasiBintang;

  /* Centaurus = 0 .. Perseus =5, Andromeda = 31, Cassiopea 32 */
```

Contoh pemakaian:

```
j:=99; ff:= 0.5;

i++; rf = rf*5.0; numero = three; RasiBintang = Centaurus;
```

Catatan:

- Objek bertipe void tidak dapat didefinisikan, karena ukuran tidak diketahui. Tipe ini dipakai untuk return value fungsi atau untuk mendeklarasikan/mendefinisikan pointer ke objek yang tipenya tidak diketahui.

- C tidak melakukan pemeriksaan saat *run time* untuk memastikan bahwa harga yang di-assign ke variabel sesuai dengan range yang ada.

PEMBAHASAN

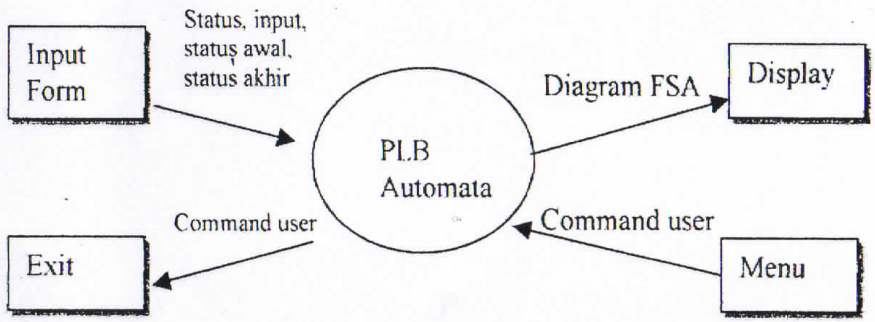
1 SPESIFIKASI SISTEM

Perangkat lunak bantu ini memiliki spesifikasi sebagai berikut :

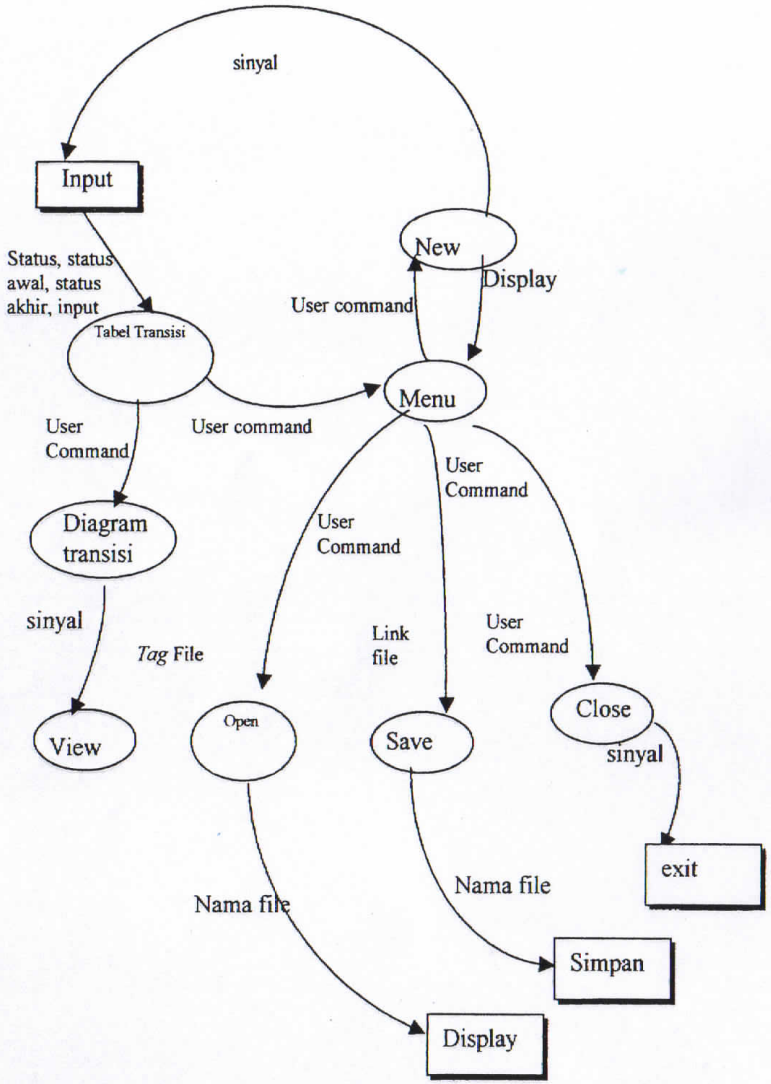
1. Membaca input berupa masukan masukan jumlah input dan status-statusnya .
2. Membuat diagram transisi dari masukan masukan tadi.
3. Membuat fasilitas menyimpan dari data-data yang diinputkan
4. Hanya menangani untuk 10 buah status dan 10 buah masukan.
5. Hanya mengenal / membuat rancangan diagram transisi FSA, sehingga belum bisa membedakan varian-varian FSA seperti *NFA, DFA, NFA* transisi ε atau *FST* apalagi ekivalensi antara *NFA* dan *DFA*, Mesin *Moore* dan Mesin *Mealy*

2. ANALISIS DATA

Berikut adalah diagram alir data (DAD) level 0 (gambar 4) dan level 1 (gambar 5) :



Gambar 4 DFD level 0



Gambar 5 DFD level 1

3. RANCANGAN FUNGSI

Fungsi-fungsi utama yang dirancang dikaitkan dengan perintah-perintah yang disediakan, yaitu:

New, untuk membuat file baru yang berisi data-data FSA

Open, untuk membuka suatu file dan menampilkan data-data yang telah ada

Save, untuk menyimpan file yang memuat data data FSA

Close, untuk menutup file yang sedang dibuka dan keluar dari sistem

Diagram Transisi, untuk menampilkan diagram rancangan FSA

Hapus semua sel, untuk menghapus daftar tabel transisi dan dari sini dapat dibuat tabel transisi yang baru

4. RANCANGAN STRUKTUR DATA INTERNAL

Ada dua struktur data utama pada program, yang masing-masing berfungsi menyimpan informasi mengenai:

4.1. Pengelolaan Diagram

record diagram dan tabel diagram, yang berfungsi untuk menampung posisi posisi dari himpunan status, status awal, maupun status akhir. Struktur ini dipilih karena sederhana dalam pengelolaannya dan sesuai untuk menampung hubungan antar status dan inputnya. Penamaan status sudah ditetapkan yaitu ada maksimum 10 buah status yang masing-masing diberi q0, q1, q2, q3, q4, q5, q6,q7, q8,q9.

4.2. Pengelolaan Legenda

Struktur data untuk pengelolaan legenda adalah sebuah record yang berfungsi untuk menampung warna legenda dan nama input berdasarkan warna legenda. Struktur ini dipilih karena paling sesuai untuk menampung *property* legenda yang terdiri dari 2 field.

5. PENGUJIAN

KASUS 1:

Suatu FSA dengan defenisi sebagai berikut :

Q = {q0,q1,q2}

Σ = {a, b}

S = {q0}

F = {q2}

Fungsi transisi sebagai berikut :

δ(q0,a) = q0

δ(q0,b) = q1

δ(q1,a) = q1

δ(q1,b) = q2

δ(q2,a) = q1

δ(q2,b) = q2

KASUS 2:

Suatu FSA dengan defenisi sebagai berikut:

Q = {q0,q1,q2}

Σ = {a,b}

S = {q0}

F = {q0}

Fungsi transisi sebagai berikut :

δ(q0, a) = q0

δ(q0, ε) = q1

δ(q1, b) = q2

δ(q2, ε) = q0

δ(q2,b) = q2

KASUS 3:

Suatu FSA dengan defenisi sebagai berikut:

Q = {q0,q1,q2,q3,q4,q5}

Σ = {0,1}

S = {q0}

F = {q3, q4}

Fungsi transisi sebagai berikut :

δ(q0, 0) = q1

δ(q0, 1) = q2

δ(q1, 0) = q2

δ(q1, 1) = q3

δ(q2, 0) = q2

δ(q2, 1) = q4

δ(q3, 0) = q3

δ(q3, 1) = q3

δ(q4, 0) = q4

δ(q4, 1) = q4

δ(q5, 0) = q5

δ(q5, 1) = q4

Kasus 4

Diberikan sebuah vending mesin dengan spesifikasi formal FSAnya sebagai berikut :

Q = {q0,q1,q2,q3,q4}

Σ = {0,1}

S = {q0}

F = {q2, q3, q4}

Fungsi transisi sebagai berikut :

δ(q0, 0) = q1,q2

δ(q0, 1) = q2

δ(q1, 0) = q3

δ(q1, 1) = q3,q4

δ(q2, 0) = q2

δ(q2, 1) = q4

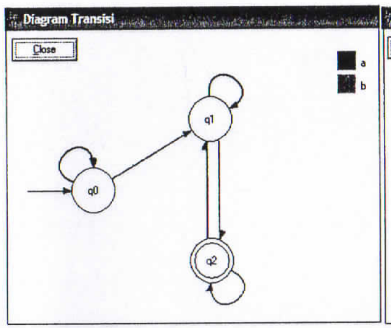
δ(q3, 0) = q3,q2

δ(q3, 1) = q3

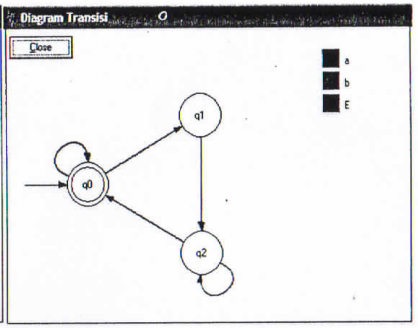
δ(q4, 0) = q4

δ(q4, 1) = q4

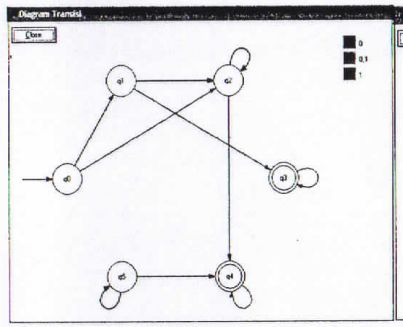
Diagram transisinya berdasarkan PLBPBO dapat dilihat pada gambar 6, gambar 7, gambar 8, dan gambar 9 sebagai berikut :



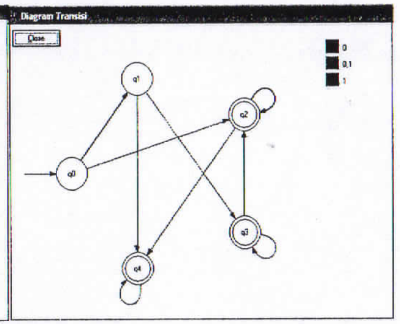
Gambar 6 Diagram FSA untuk kasus 1



Gambar 7 Diagram FSA untuk kasus 2



Gambar 8 Diagram FSA untuk kasus 3



Gambar 9 Diagram FSA untuk kasus 4

Gambar diatas menunjukkan hasil sesuai dengan yang diharapkan.

PENUTUP

Perangkat lunak perancangan berbasis otomata yang dikembangkan di sini adalah sebuah perangkat lunak sederhana dengan kemampuan hanya mampu memdesain FSA saja, tidak termasuk varian-variannya. Struktur data yang dipakai untuk membangun perangkat lunak ini juga struktur data yang sederhana yaitu record, array dan enumerasi. Alasan pemilihan struktur ini adalah karena akan membuat pemakaian memori menjadi efektif dan efisien sesuai dengan kebutuhan sistem.

DAFTAR PUSTAKA

1. Afergan, M. & friends , *Web Programming Desktop Reference 6-IN-1*, Que Corp., 1996
2. Kernighan B.W., Ritchie D.M., *The C Programming Language*, Prentice Hall, Second Edition, 1988.
3. Kelley, D., *Otomata dan Bahasa Bahasa Formal*, Prenhallindo, 1999
4. Liem I., *Diktat Kuliah IF 223 Algoritma dan Pemrograman*, Jurusan Teknik Informatika ITB, 1999
5. Liem I., *Diktat Struktur Data*, Jurusan Teknik Informatika ITB, 2001
6. Pressman R.S., Software Engineering A Practitioner's Approach, Mc Graw-Hill 1997.
7. Utdirartatmo, F., *Teori Bahasa dan Otomata*, J&J Learning, 2001
8. Wirth, N., *Algorithms & Data Structures*, Prentice Hall, 1986